GB04/02471

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

REC'D 13 JUL 2004
WIPO PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

I also certify that the application is now proceeding in the name as identified herein.

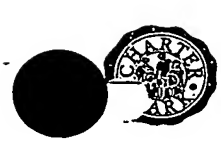
In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed Stephen Hordley

Dated 1 July 2004



INVESTOR IN PEOPLE

GB 0313375.8

By virtue of a direction given under Section 30 of the Patents Act 1977, the application is proceeding in the name of:

SYMBIAN SOFTWARE LTD,
2-6 Boundary Row,
Southwark,
LONDON,
SE1 8HP,
United Kingdom

Incorporated in the United Kingdom,

[ADP No. 08843435001]

For Official use only

THE PATENT OFFICE
C
10 JUN 2003
LONDON

11JUN03 E813949-1 D10092
P01/7700 0.00-0313375.8

Your reference **Con Serv Broker (UK)**

The **Patent Office**

Request for grant of a Patent

Form 1/77 Patents Act 1977

1 Title of invention

Method of connecting a client running on a computing device to a server running on a different computing device

2. Applicant's details 10 JUN 2003 0313375.8

[X] First or only applicant

2a If applying as a corporate body: Corporate Name

Symbian Limited

SECTION 20 (1977 ACT) APPLICATION FILED 10/6/04

Country

GB

2b If applying as an individual or partnership

Surname

Forenames

2c Address

Sentinel House
16 Harcourt Street
London

UK Postcode W1H 1DS

Country GB

ADP Number 07573132002

☐	Second applicant (if any)	
2d	Corporate Name	
	Country	
2e	Surname	
	Forenames	
2f	Address	
	UK Postcode	
	Country	
	ADP Number	
3	Address for service	
	Agent's Name	Origin Limited
	Agent's Address	52 Muswell Hill Road London
	Agent's postcode	N10 3JR
	Agent's ADP Number	C03274

07270457002

4 Reference Number

Con Serv Broker (UK)

5 Claiming an earlier application date

An earlier filing date is claimed:

Yes ☐ No ☒

Number of earlier application or patent number

Filing date

15 (4) (Divisional)	8(3)	12(6)	37(4)
☐	☐	☐	☐

6 Declaration of priority

Country of filing	Priority Application Number	Filing Date

7 Inventorship

The applicant(s) are the sole inventors/joint inventors

Yes [ ] No [X]

8 Checklist

		Continuation sheets	
Claims	3	Description	16 RM
Abstract	0	Drawings	2+2

Priority Documents	~~Yes~~/No
Translations of Priority Documents	~~Yes~~/No
Patents Form 7/77	~~Yes~~/No
Patents Form 9/77	~~Yes~~/~~No~~
Patents Form 10/77	~~Yes~~/No

9 Request

We request the grant of a patent on the basis of this application

Signed: Origin Limited Date: 9 June 2003

(Origin Limited)

METHOD OF CONNECTING A CLIENT RUNNING ON A COMPUTING DEVICE TO A SERVER RUNNING ON A DIFFERENT COMPUTING DEVICE

BACKGROUND TO THE INVENTION

1. Field of the Invention

This invention relates to a method of connecting a client running on a computing device to a server running on a different computing device.

2. Description of the Prior Art

When a client (i.e. a program making a request for a service) wishes to connect to a server (i.e. the program that can supply the requested service) over a network, it needs to uniquely identify the service. The standard approach relies on the use of 'port numbers'. In essence, a port number is a logical address: when one program communicates with another program on a different computer, it specifies the port number of that program in each data transmission so that its messages can be properly received by the correct program. For instance, HTTP normally uses port number 80: all HTTP messages from a client are uniquely identified by the client specifying port 80.

This approach requires manual configuration (i.e. the developer of the client program has to manually select a port number, with ICANN designating the so-called 'well-known' port numbers 0 to 1023; registered ports running from 1024 to 49151 and private ports running from 49152 to 65535). It also risks clashes between servers (i.e. if two developers choose the same port number). Nevertheless, this approach is workable for services provided by OS or device manufacturers because those port numbers can be fixed when a device is manufactured. However, Independent Software Vendors (ISVs) cannot reserve these port numbers and so the conventional approach makes it difficult for ISVs to create new services since they face the risk of port number allocations being duplicated, with the risk of clashes arising.

SUMMARY OF THE PRESENT INVENTION

In an implementation of the invention, services installed on a computing device register their published name, which conforms to a structured naming convention, such as reversed domain information, with a 'service broker' on that device. The service broker uses a single well-known port number address. When an external client wants to use a service on the connected computing device, it sends a message to the service broker using the well known port number. The message specifies the name of the desired server and requests that the service broker inform it of the appropriate connection point (e.g. port number) to use. There is no dependency on port numbers or unstructured and arbitrary naming conventions.

Because service names are made unique by using a structured (and preferably standard and open) naming convention, e.g. by pre-pending the service name with reversed domain information, new connectivity services can be added to devices without having to change the existing configuration of the device. Address clashes are avoided as long as new services use the service broker and a consistent naming convention.

If a service is registered with the name specified by a client to the service broker, then the service broker starts the service. The service obtains a connection point by some means and informs the service broker of the connection point address (for TCP/IP this would be a port number, other transport mechanisms (e.g. Bluetooth, serial, USB, IrDA etc.) would have other addressing mechanisms). The service broker then informs the external client of the connection point address of the service. The client then communicates directly with the server, unlike some prior art approaches, where the intermediary stays in position. With this mechanism the only statically allocated address is that of the service broker.

If a service is required more than once, the server will not be re-started: instead the service broker uses cached address information.

When services register with the service broker, they may register a version number to indicate the version of the service that they are providing (and, optionally, other information and how they can be started). The external client can request a specific version of a named service or it can omit the version, in which case the service broker will start the highest version available of the named service. In either case, the service broker informs the remote client of the version of the service that has been started. This allows a remote client to inter-operate with a range of devices and potentially handle different versions of services.

The service broker can serve external clients that are PC's or other computers connected by a local link such as cable, Infra-Red or short distance radio (such as Bluetooth) or by a remote link such as a network data connection.

The service broker can provide authentication information such that only authenticated external clients can access services.

The service broker does not require administration, it is extensible and resolves port clashing by using names instead of numbers and names can be easily made unique by using internet domain information.

It provides mechanisms for handling and publishing service version numbers and thus allows one client to handle a range of devices.

Appendix 1 describes a detailed implementation of the present invention for Symbian OS.

Appendix 1

Service Broker Functional Specification

Security Classification:	Confidential
Document Reference:	SGL.GT0170.058
Status:	Draft
Version:	0.02
Last Revised Date:	9th May 2003
Team/Department :	PC Connectivity, Software Engineering
Author(s):	Stefania Alborghetti
Owner(s):	PC Connectivity Team

Approver(s):	PC Connectivity Team	**Distribution:**	PC Connectivity Team, Licensees

Introduction

Purpose and Scope

The purpose of this document is to describe the functionality provided by the Service Broker.

The Service Broker provides a mechanism for remote clients to start various service providers[1] on the device and to retrieve the port number for these services. It also manages connection authentication, between the device and a remote client. Remote clients are requested to authenticate the connection before starting named services on the device.

Overview Context

Error! Reference source not found. shows the Service Broker in its architectural context. The Bearer Abstraction Layer (BAL) running on the remote client connects to the server socket provided by the Service Broker. The remote client is usually a Windows PC but it can be any device capable of establishing a TCP/IP connection to the device.

An application on the remote client requests the service port number from the BAL (defined in [3]). The BAL requests this port number from the Service Broker via the messaging protocol described in [1]. The name of the service is specified by the client to the BAL and by the BAL to the Service Broker.

The Service Broker attempts to start the Service Provider (if not already running) and to retrieve its port number. This is achieved via the Client Server API described in [2].

Once the Service Provider has been started, the Service Broker transmits the port number to the BAL, which in turn communicates it to the client application. The client application can then establish a direct connection to the Service Provider.

Functionality

Functionality for the Service Broker is expressed as a set of use cases, which are triggered by the external interfaces.

---

[1] Socket servers using the TCP/IP protocol suite and communicating via a protocol that must have been defined between client and server.

table. There is one entry for each service containing the following:-

- Service Name;
- Process Name (the name of the process that implements this service);
- Exe Location (full path and file name for the exe file implementing the process).
- Command line arguments (the command line arguments to be used when starting the process).
- Service Version (minor, major and build number).
- Registration file name (the name of the registration file that specifies the service name)[3].

Normal Execution Flow

If starting up scan the ROM registration directory. For every xml file (extension .xml) perform as specified in 0.

Scan the non-ROM registration directory.

For every xml file that has been removed:

The services belonging to the file are removed from the table.

For every file that has been added or updated:-

Open the file and parse it.

Store the service into the services table. There can be more than one service defined in a registration file.

Close the file

Post-condition

The services table is up to date with the registration directory.

Exceptions

A registration file cannot be opened, in this case log an error and ignore the registration file.

The syntax of a registration file is wrong, in this case log an error and ignore

[3] This is not specified in the file itself but it is stored in the table because a service can be overwritten later on only if it is in the same registration file. Also, if registration files are deleted, the services corresponding to this file will be removed from the table.

	the registration file.
	A registration file specifies a service name, which is already registered. In this case, only update the services table if the registration file is the same, otherwise log an error and ignore this service.
	A registration file specifies a service name more than once. In this case override the previous service information, only keep the one read last. However, log an error message.

Shutdown

Pre-condition	The System is running.
Description	The System is asked to terminate, this is usually done during the device shutdown procedure.
Normal Execution Flow	Stop the client server interface disconnecting any active sessions.
	Disconnect all the connected remote clients. Close the server socket.
	Terminate execution.
Post-condition	The system has terminated.
Exceptions	None.

Accept Remote Client Connection

Pre-condition	"Startup" has completed successfully.
Description	Accept a connection from a remote client. The maximum number of client connections that can be accepted at the same time is defined in [1].
Normal Execution Flow	Create a new client socket.
	Receive messages on the socket.
Post-condition	There is a new client socket.
Exceptions	The client socket cannot be created. In this case the client connection request is not accepted and an error is displayed.
	An unrecognised message is received, in this case discard the message and

log and error..

Get Device Information

Pre-condition	"Accept Remote Client Connection" has completed successfully.
Description	A "Get Device Information" message has been received from the remote client. Messages are described in [1].
Normal Execution Flow	Read the message header. Retrieve the device information (from interfacing to the HAL). Compose "Device Information" and send it to the client. The message ID of the outgoing message must be the same as the one received in the incoming message.
Post-condition	The message has been handled.
Exceptions	The device information cannot be retrieved for some reason. In this case, compose an error message and send it to the client. The error code is "Internal Failure"[4].

Get Version

Pre-condition	"Accept Remote Client Connection" has completed successfully.
Description	A "Get Version" message has been received from the remote client.
Normal Execution Flow	Read the message header. Compose "Device Version" and send it to the client. The message ID of the outgoing message must be the same as the one received in the incoming message.
Post-condition	The message has been handled.
Exceptions	None.

[4] Error codes are defined in [1].

Authenticate Connection

Pre-condition	"Accept Remote Client Connection" has completed successfully. The connection is not authenticated (see 0).
Description	A "Authenticate Connection" message has been received from the remote client.
Normal Execution Flow	Read the message header and generate a random number. Send the random number to the client in a "Random Value" message. Wait for a "Password" message from the client. When "Password" is received from the client communicate the random number to the password provider DLL and retrieve a hash of the password and the random number. Compare this hash with the one received from the remote client in the "Password" command. If the two hashes are the same then send a "Connection Authenticated" message to the client. This indicates that any connection received from the client (from the same IP address and on the same physical bearer) should be considered authenticated until the physical link is dropped.
Post-condition	The connection is authenticated.
Exceptions	If the hash supplied by the client is not the same as the hash created by the password provider DLL send an error message to the client, with code "Authentication Failed". If the hash cannot be retrieved from the password provider DLL, send an error message to the client with code "Authentication Failed" and log an error message as well.

Get Services

Pre-condition	"Accept Remote Client Connection" has completed successfully. The connection is authenticated.
Description	A "Get Services" message has been received from the remote client.

Normal Execution Flow	Read the message header.
	Retrieve the list of services from the services table (0).
	Compose "Services" and send it to the client.
	The message ID of the outgoing message must be the same as the one received in the incoming message.
Post-condition	The message has been handled.
Exceptions	No services are available. In this case return a "Services" message with an empty list.
	The connection is not authenticated, in this case return to the client an error message with code "Not Authenticated".

Get Service Version

Pre-condition	"Accept Remote Client Connection" has completed successfully. The connection is authenticated.
Description	A "Get Service Version" message has been received from the remote client.
Normal Execution Flow	Read the message header and extract the service name from the message data.
	Retrieve the version supported for this service from the services table (0).
	Compose "Service Version" and send it to the client.
	The message ID of the outgoing message must be the same as the one received in the incoming message.
Post-condition	The message has been handled.
Exceptions	The specified service is not in the services table. In this case send an error message to the client with error code "Not Found".
	The connection is not authenticated, in this case return to the client an error message with code "Not Authenticated".

Start Service

Pre-condition	"Accept Remote Client Connection" has completed successfully. The connection is authenticated.
Description	A "Start Service" message has been received from the remote client.
Normal Execution Flow	Read the message header and extract the service name. Also extract a timeout value as specified in [1].
	Check if the process associated with this service (the Service Provider) is already running. Use the information stored in the services table to do this (0).
	If the process is not running start the process and a timer using the timeout extracted above.
	When the Service Provider registers itself (0), send its port number to the client in a "Service Started" message and stop the timer.
	At the expiry of the timer send an error to the client with code "Failed to Register".
	The message ID of the outgoing message must be the same as the one received in the incoming message.
	If the Service Provider is already running then compose "Service Started" and send it to the client. The port number is stored in the services table.
	The message ID must be the same as the one received in the incoming message.
Post-condition	The message has been handled and the requested Service Provider is running.
Exceptions	The specified service is not in the services table. In this case send an error message to the client with code "Not Found".
	The process cannot be started. In this case send an error message to the client with error code "Cannot Start".

When registering itself, the process may communicate a failure code instead of a port number. This can be done via the API specified in [2]. In this case send an error message to the client with error code "Cannot Start" and with extended error code equal to the error received from the process.

The connection is not authenticated, in this case return to the client an error message with code "Not Authenticated".

Disconnect remote client

Pre-condition	"Accept Remote Client Connection" has completed successfully. The client closes the socket or an unrecoverable error occurs or execution is terminated for the entire system.
Description	The client connection is closed.
Normal Execution Flow	Close the socket connection and release any resources.
Post-condition	The client has been disconnected.
Exceptions	None.

Accept Service Provider Connection

Pre-condition	"Startup" has completed successfully.
Description	A new client session is established.
Normal Execution Flow	Establish a new session with the client.
Post-condition	A new client is connected to the Client Server Interface.
Exceptions	None.

Register Service Provider

Pre-condition	"Accept Service Provider Connection" has completed correctly. "Start Service" may be ongoing (0), but not necessarily.
Description	The Service Provider communicates the port number corresponding to the specified service name.

Normal Execution Flow	Retrieve the service name and port number from the Service Provider. Verify that the executable file of the client (full path and file name) is the same as the exe in the services table for the specified service name (0). Store the port number in the services table. If one or more client requests are pending for this service, send the service port number to the clients as specified in 0 for all the pending requests[5].
Post-condition	The port number has been added to the services table.
Exceptions	The service name is not found in the table, in this case return an error. This can happen if the service name has no registration file associated with it (0). The service name already has a port number assigned to it. In this case override the port number with the new value and log an error. This can happen when a named service is started, then it terminates, then it is started again and so forth. Instead of communicating a port number, the Service Provider communicates a startup failure code. This is possible using the API specified in [2]. In this case update the services table with the failure code and set the port number to 0 (invalid port number).

Disconnect Service Provider

Pre-condition	"Accept Service Provider Connection" has completed successfully.
Description	The client session terminates.
Normal Execution Flow	Close the client session and release any resources associated with it (except for the information stored in the services table).
Post-condition	The client has been disconnected from the client server interface.
Exceptions	None

[5] Note however that a client may only have one pending request at a time.

Further Information

People

Role	Person / People
Contributor(s)	Stefania Alborghetti
Reviewer(s)	PC Connectivity team

Glossary

The following technical terms and abbreviations are used within this document.

Term	Definition
API	Application Programming Interface
BAL	Bearer Abstraction Layer
HAL	Hardware Abstraction Layer
IP	Internet Protocol
TCP	Transport Control Protocol

Document History

Date	Version	Status	Description
17-04-2003	0.01	Draft	Initial version
09-05-2003	0.02	Draft	Review of version 0.01

References

Num	Version	Reference	Description
[1]	0.02	SGL.CDV0000.109	Service Broker Protocol Specification
[2]	0.02	SGL.GT0170.059	Service Broker API Specification
[3]	0.01	SGL.GT170.050	Bearer Abstraction Layer (BAL) Description

CLAIMS

1. A method of connecting a client running on a first computing device to a server running on a second computing device, comprising the steps of:

(a) a service installed on the second computing device registering its published name, with a service broker on that device;

(b) the client sending a message to the service broker specifying the name of the service;

wherein the published name of the service conforms to a structured naming convention.

2. The method of Claim 1 in which the structured naming convention uses reversed domain information.

3. The method of Claim 1 in which the service broker uses a single well-known port number address.

4. The method of Claim 1 in which the service obtains a connection point and informs the service broker of the connection point address and the service broker then informs the client of the connection point address.

5. The method of Claim 4 in which the service broker informs the client of the connection point address and the client then uses that address in communicating directly with the server.

6. The method of Claim 4 in which the connection point address is a port number.

7. The method of Claim 4 in which, if a service is required more than once, the server providing the service will not be re-started, but instead the service broker uses cached address information.

8. The method of Claim 1 in which, when services register with the service broker, they register a version number to indicate the version of the service that they are providing.

9. The method of Claim 8 in which the client can request a specific version of a named service or it can omit the version, in which case the service broker will start the highest version available of the named service.

10. The method of Claim 1 in which the service broker can serve external clients that are PC's or other computers connected by a local link such as cable, Infra-Red or short distance radio (such as Bluetooth) or by a remote link such as a network data connection.

11. The method of Claim 1 in which the service broker provides authentication information such that only authenticated external clients can access services.

12. A computing device comprising (a) a server that connects to a first computing device, and (b) a service broker to which a service installed on the computing device registers its published name and which receives a message from the first computing device specifying that name; wherein the published name of the service conforms to a structured naming convention.

13. The device of Claim 12 in which the service broker is programmed such that the structured naming convention uses reversed domain information.

14. The device of Claim 12 in which the service broker uses a single well-known port number address.

15. The device of Claim 12 in which the service obtains a connection point and informs the service broker of the connection point address and the service broker then informs the client of the connection point address.

16. The device of Claim 15 in which the service broker informs the client of the connection point address and the client then uses that address in communicating directly with the server.

17. The device of Claim 15 in which the connection point address is a port number.

18. The device of Claim 15 in which, if a service is required more than once, the server providing the service will not be re-started, but instead the service broker uses cached address information.

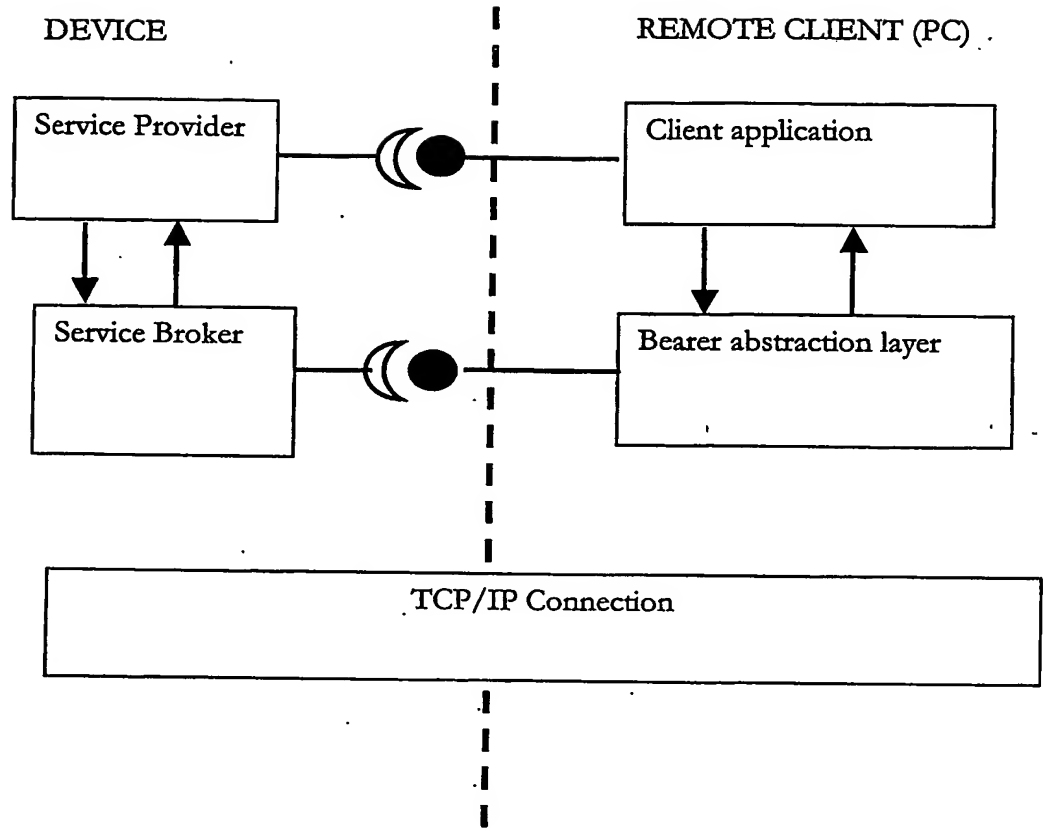
19. The device of Claim 12 in which, when services register with the service broker, they register a version number to indicate the version of the service that they are providing.

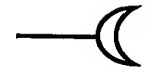
20. The device of Claim 19 in which the client can request a specific version of a named service or it can omit the version, in which case the service broker will start the highest version available of the named service.

21. The device of Claim 12 in which the service broker can serve external clients that are PC's or other computers connected by a local link such as cable, Infra-Red or short distance radio (such as Bluetooth) or by a remote link such as a network data connection.

22. The device of Claim 12 in which the service broker provides authentication information such that only authenticated external clients can access services.

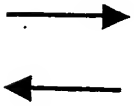
Figure 1





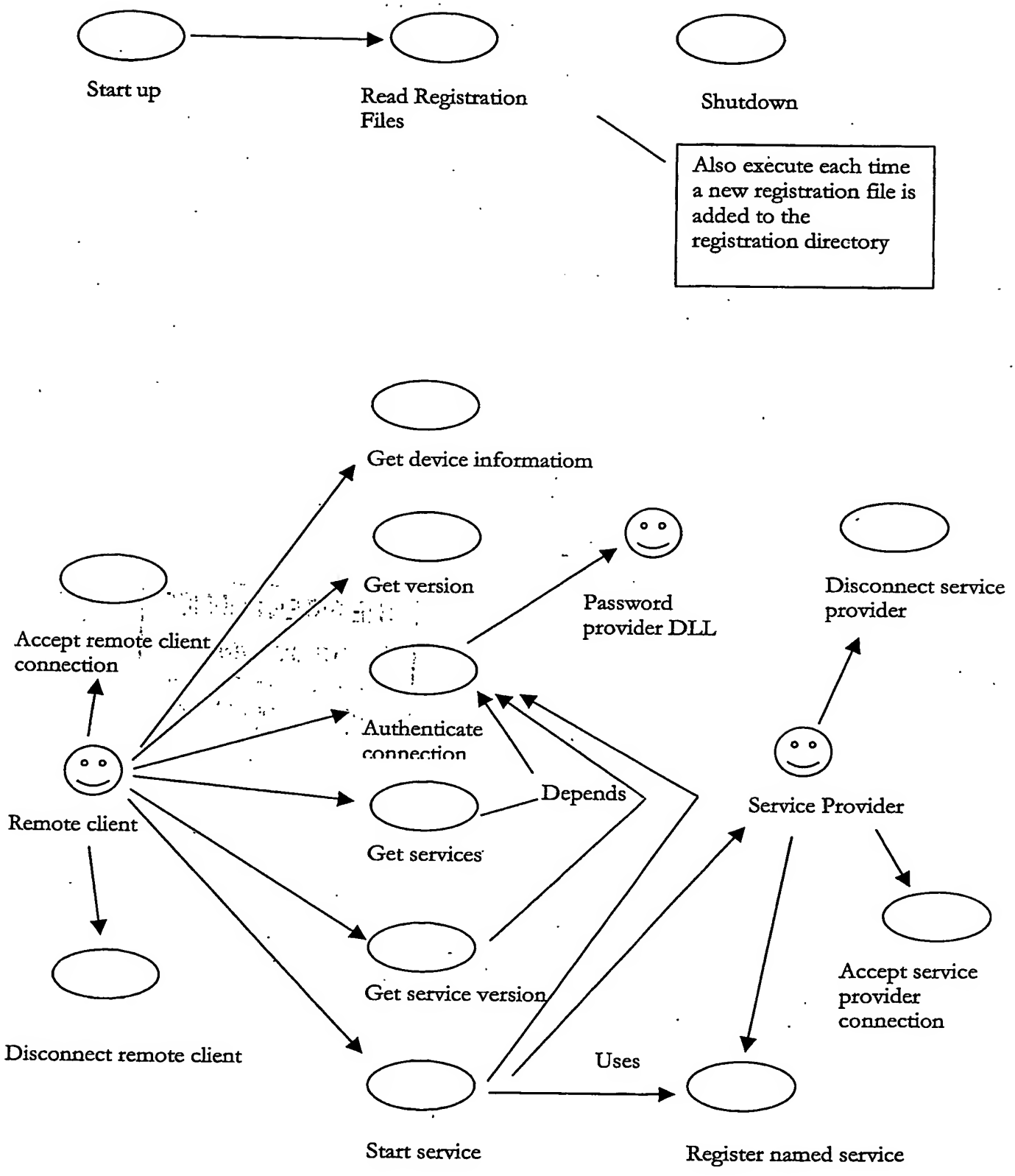
Server socket connection

Client socket connection



Internal interface (SCOM on the PC or Client/Server on the device

Figure 2


Start up
Read Registration Files
Shutdown
Also execute each time a new registration file is added to the registration directory
Get device informatiom
Get version
Password provider DLL
Disconnect service provider
Accept remote client connection
Authenticate connection
Depends
Service Provider
Remote client
Get services
Get service version
Accept service provider connection
Disconnect remote client
Uses
Start service
Register named service

PCT/GB2004/002471